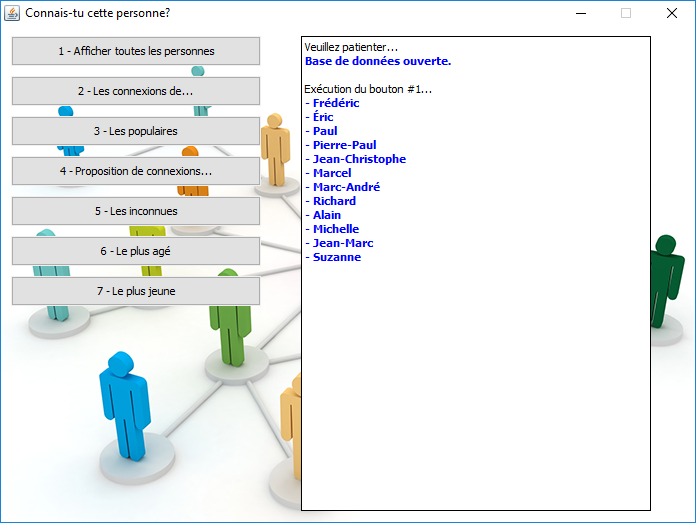
# **Introduction à Neo4j**

L’objectif de ce laboratoire est de s’initier à une base de données de type graphe.

## L’application « Connais-tu cette personne ? »

Ce programme permet simplement d’interroger une base de données graphe existante.



## Notes

* Pour réaliser ce laboratoire, je vous recommande FORTEMENT de suivre les étapes en ordre.
* Si vous avez des questions, n’hésitez pas à me demander de l’aide.

## Exemple d’opérations pour Neo4j dans l’application.

### Lecture d’une liste

Dans le UserDAO :

List<String> nomsDesProduits =

executerRequeteCypher("MATCH (produit) WHERE produit.prix > 300 RETURN produit.nom");

### Lecture d’une liste avec un paramètre

IMPORTANT : Dans ce projet, le paramètre doit s’appeler « {p1} ».

List<String> listeEntrepots =

executerRequeteCypher("MATCH (p) WHERE p.nom = $p1 RETURN p.endroit", "xbox");

### Lecture avec relation

Retourne la liste de ce qui est inclut lors de l’achat d’un XBOX.

List<String> result =

executerRequeteCypher("MATCH (a)-[:CONTIENT]->(b) WHERE a.name = 'xbox' RETURN b.name");

### Vérifier l’absence d’une relation (NOT …)

Cette liste retourne le produit le plus dispendieux.

List<String> result =

executerRequeteCypher("MATCH (produit) WHERE NOT(()-[:PLUS\_DISPENDIEUX]->(produit)) RETURN

DISTINCT produit.nom")

## Les tâches à réaliser

### Tâche #0 : Vider votre base de données

Supprimez tous les nœuds de votre base de données (via une requête Cypher)

#### Notes/Énoncés :

MATCH (n)

DETACH DELETE n

### Tâche #1 : Afficher la liste des personnes de la BD

Faites afficher le nom des personnes présentes dans la base de données ordonnées selon leur nom.

Méthode à compléter : UserDAO.findAllUsers()

Réponse : Le nom de 12 personnes ordonnées selon le nom

#### Notes/Énoncés :

result = executerRequeteCypher("MATCH (Person) RETURN Person.name ORDER BY Person.name LIMIT 12");

### Tâche #2 : Les connaissances d’une personne

Pour un usager, retourner toutes les personnes qu'il connait (la relation est « KNOWS ») ordonnées selon leur nom.

Méthode à compléter : UserDAO.getDirectConnectionsOf (nom)

Réponse :

* Si vous écrivez « Marcel », la réponse est : Éric
* Si vous écrivez « Paul », la réponse est : Frédéric, Marcel

#### Notes/Énoncés :

result = executerRequeteCypher("MATCH (person: Person)-[:KNOWS]->(freinds: Person) WHERE person.name = $p1 RETURN freinds.name", user);  
  
result = executerRequeteCypher("MATCH (person: Person {name: $p1})-[:KNOWS]->(friends) RETURN friends.name", user);

### Tâche #3 : Les populaires

Retourner la liste des personnes qui sont connues de 2 personnes ou plus, ordonnées selon le nom.

Méthode à compléter : UserDAO.getPopularUsers ()

Réponse : Frédéric, Marcel, Paul, Richard, Éric

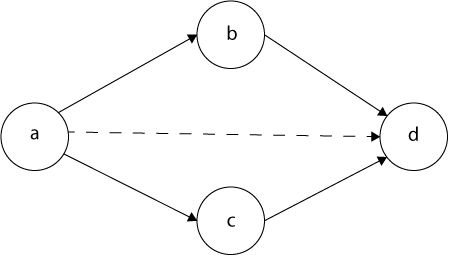
#### Notes/Énoncés :

String query = "MATCH (person:Person)<-[:KNOWS]-(friend) " +  
 "WITH person, COUNT(friend) AS friendsCount " +  
 "WHERE friendsCount >= 2 " +  
 "RETURN person.name " +  
 "ORDER BY person.name";  
result = executerRequeteCypher(query);

### Tâche #4 : Proposition de connexions

Si deux contacts directs d'une personne connaissent la même personne, mais que celui-ci ne la connait pas... pourquoi ne pas lui proposer ?

Autrement dit (**a** ne connait pas **d**):



Méthode à compléter : UserDAO.proposeConnection(nom) 🡨 ordonnez selon le nom

Réponse :

* Si vous spécifiez « Marc-André », cela devrait donner : Frédéric, Marcel

#### Notes/Énoncés :

String query = "MATCH (user:Person {name: $p1})-[:KNOWS]->(friend)-[:KNOWS]->(friendOfFriend) " +  
 "WHERE NOT (user)-[:KNOWS]->(friendOfFriend) AND user <> friendOfFriend " +  
 "WITH friendOfFriend, COUNT(DISTINCT friend) AS mutualConnections " +  
 "WHERE mutualConnections >= 2 " +  
 "RETURN friendOfFriend.name AS suggestedFriend " +  
 "ORDER BY friendOfFriend.name";  
result = executerRequeteCypher(query, user);

### Tâche #5 : Les inconnues

Certaines personnes ne sont connues de personnes... qui ?

Méthode à compléter : UserDAO.checkUnconnected () 🡨 ordonnez selon le nom

Réponse : Alain, Jean-Marc, Marc-André, Michelle, Pierre-Paul, Suzanne

#### Notes/Énoncés :

String query = "MATCH (person:Person) " +  
 "WHERE NOT (()-[:KNOWS]->(person)) " +  
 "RETURN person.name AS UnconnectedPerson " +  
 "ORDER BY person.name";  
result = executerRequeteCypher(query);

### Tâche #6 : La plus âgée ?

Quelle est la personne la plus âgée ? (la relation est OLDER\_THAN)

Méthode à compléter : UserDAO.getOldest()

Réponse : Marcel

#### Notes/Énoncés :

result = executerRequeteCypher("MATCH (person:Person) WHERE NOT (:Person)-[:OLDER\_THAT]->(person) RETURN person.name");